

A Hybrid Deep-Boosting Framework for Context-Aware Severity Classification in Healthcare Security Narratives

P. Anupama¹, Pavithra¹, Jangam Kailash¹, Siliveru Kishan¹

¹Department of Computer Science and Engineering, ¹Sree Dattha Institute of Engineering and Science,
Nagarjuna Sagar Road, Sheriguda, Ibrahimpatnam, Rangareddy Dist, 501510, Telangana, India.

Abstract

Healthcare infrastructures are becoming prime targets for cyber threats, as evidenced by the massive exposure of patient records and the continuous rise in reported vulnerabilities within hospital systems. The growing dependence on digital platforms has intensified the need for efficient mechanisms to detect and prioritize security issues. Traditional manual methods for assessing bug severity are not only time-intensive but also inconsistent, particularly when dealing with large volumes of unstructured security reports generated in medical environments. To address these challenges, this research introduces an intelligent Natural Language Processing (NLP)-based framework designed to automate bug severity classification. The approach utilizes a specialized dataset containing medical incident reports, security advisories, and vulnerability disclosures. Initially, the data undergoes systematic preprocessing and Exploratory Data Analysis to ensure cleanliness, consistency, and meaningful representation. For capturing contextual semantics, a lightweight transformer-based model inspired by Robustly Optimized bidirectional encoded representation of transformers (RoBERTa) is employed, enabling efficient generation of text embeddings with reduced computational overhead. To further enhance predictive performance, the framework integrates a Deep Neural Network (DNN) for feature refinement, which identifies and extracts the most relevant patterns from the embedding space. These optimized features are then passed to a probabilistic boosting-based classifier, improving classification reliability compared to conventional standalone models. The system categorizes vulnerabilities into two classes normal and severe facilitating rapid prioritization of critical issues. The framework demonstrates improved accuracy, reduced misclassification, and faster processing capability. By combining efficient language modelling with advanced feature learning, this approach offers a scalable and practical solution for strengthening cybersecurity management in healthcare environments.

Keywords: Healthcare Cybersecurity, Bug Severity Classification, Vulnerability Detection, Natural Language Processing (NLP), Feature Extraction, Deep Neural Networks (DNN)

1. Introduction

As software systems continue to grow in scale and complexity, ensuring their reliability has become increasingly challenging. Even well-designed applications are susceptible to defects that arise during development, often due to coding errors, improper resource handling, or unexpected runtime conditions. These defects, commonly referred to as software bugs, can compromise system functionality and, in critical environments, lead to severe security and operational consequences. In domains where reliability is essential such as healthcare, finance, and communication systems even minor faults can escalate into major disruptions, as shown in figure 1. Effective management of software bugs depends not only on detecting them but also on accurately determining their severity. In typical workflows, users or testers submit bug reports through tracking systems, including details such as descriptions, system context, and perceived impact.

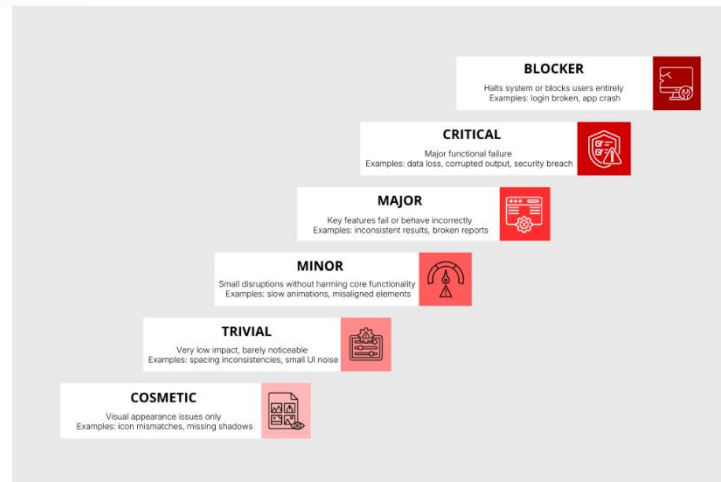


Figure. 1: Bug severity in medical data.

These reports are then reviewed by developers or triagers, who must analyze large volumes of submissions and assign appropriate severity levels. This process is inherently subjective and often inefficient, especially when the number of incoming reports is high. Bug severity generally reflects the extent to which an issue affects system functionality. Critical defects may render an application unusable, while moderate issues degrade key features, and minor ones have limited or cosmetic impact. Misjudging severity can lead to improper prioritization, delayed fixes, and inefficient allocation of development resources. In practice, many bug reports undergo reassessment after initial classification, indicating inconsistencies in manual evaluation and contributing to longer resolution times. To address these challenges, automated bug severity classification has gained attention. Modern approaches leverage machine learning techniques to extract meaningful patterns from textual bug reports and predict severity levels more consistently. Current research focuses on improving feature representation and developing robust models capable of handling diverse and unstructured data, ultimately aiming to enhance accuracy and streamline the bug triaging process.

2. Literature Survey

Kukkar et al. [1] designed a deep learning-oriented solution for identifying bug severity by combining Convolutional Neural Networks (CNN) with a Random Forest model enhanced through boosting techniques, forming what they referred to as a BCR framework. Their approach began with detailed Natural Language Processing steps, where bug reports were cleaned and transformed into structured text. They relied on n-gram representations to capture meaningful word patterns from the reports. The CNN component was responsible for learning deep contextual features from textual data, enabling the model to understand semantic relationships more effectively. These learned features were then passed to the boosted Random Forest classifier, which improved prediction robustness. The integration of deep learning with ensemble methods allowed the system to balance feature learning and classification accuracy. Their model handled both multiclass and binary classification scenarios efficiently. It demonstrated a notable improvement in performance when compared to traditional machine learning models. The reported accuracy reached 96.34%, highlighting its strong predictive capability. This approach shows how combining multiple techniques can significantly enhance severity classification outcomes. Dao and Yang [2] introduced a comprehensive deep learning framework called MASP that focused on incorporating multiple perspectives of bug reports rather than relying on textual content alone. Their model utilized CNN architecture to extract deep features from the textual descriptions of bugs. In addition to content, they included sentiment-related features to understand the emotional tone of the report. They also considered quality metrics, which helped assess the clarity and completeness of bug descriptions. Another important

addition was reporter-related information, which captured patterns in how different users submit bug reports. By integrating these multiple aspects, the model gained a richer understanding of each report. This multi-dimensional feature representation helped improve classification performance significantly. Their method demonstrated that combining diverse contextual signals leads to better predictive outcomes. It also addressed limitations found in approaches that rely solely on text. Overall, the framework provided a more holistic way to interpret and classify bug severity. Köksal and Tekinerdogan [3] developed an automated classification approach that combines machine learning, text mining, and Natural Language Processing techniques to process bug reports written in more than one language. Their work focused on handling bilingual datasets, which are often challenging due to linguistic variations. They applied preprocessing steps to normalize and standardize text across languages. Feature extraction techniques were then used to capture relevant patterns from the reports. Their model was evaluated in an industrial setting, making the results more practical and applicable. The findings showed that automated classification reduced the workload associated with manual triaging. It also provided more consistent and accurate results compared to human-based classification. Their approach highlighted the importance of handling multilingual data in real-world applications. The system demonstrated scalability when dealing with large volumes of bug reports. This work emphasizes the benefits of automation in improving efficiency and accuracy. Albattah and Alzahrani [4] explored the effectiveness of various machine learning and deep learning techniques in predicting software defects using datasets containing numerous software metrics. Their work focused on comparing different algorithms to understand their strengths and weaknesses. They utilized large-scale datasets to ensure that the evaluation was comprehensive and reliable. Various models were tested, including both traditional and advanced learning methods. Their analysis showed that no single model consistently outperformed others in all scenarios. Instead, hybrid approaches that combine multiple techniques tended to deliver better performance. These combinations helped capture both linear and complex relationships within the data. The study also highlighted the importance of feature selection in improving prediction accuracy. Their findings provided useful insights for selecting appropriate models based on dataset characteristics. This work contributes to improving decision-making in defect prediction tasks. Qian et al. [5] presented an extensive overview of bug deduplication and triage techniques, focusing on how different methods have evolved over time. They organized existing approaches into clear categories based on the type of features used, such as textual data and runtime information. Their discussion covered a wide range of datasets commonly used in research. They also examined the strengths and limitations of various methodologies. By analyzing research trends, they provided insights into which techniques are gaining popularity. The work highlighted the growing importance of automation in handling large volumes of bug reports. It also emphasized challenges such as data imbalance and feature representation. Their structured presentation makes it easier to understand the landscape of bug triaging methods. The discussion helps identify gaps that future work can address. This contribution serves as a useful reference for understanding different classification strategies.

Tabassum et al. [6] proposed a hybrid framework that combines Natural Language Processing with machine learning techniques for multi-class bug classification and prioritization. Their approach began with preprocessing steps to clean and normalize textual data from bug reports. They used TF-IDF to capture the importance of words within documents. Additionally, word2vec embeddings were employed to represent semantic relationships between terms. These combined features provided a richer representation of the data. The model was then trained using machine learning algorithms to classify bug severity levels. Their results showed noticeable improvements after applying feature transformation techniques. The framework also demonstrated better handling of complex and unstructured text. By integrating multiple feature extraction

methods, they improved classification accuracy. This approach highlights the value of combining statistical and semantic representations in bug classification. Bhakar et al. [7] explored computational intelligence techniques for identifying severity levels in medical-related applications, focusing on conditions such as Parkinson's disease and diabetic retinopathy. Their work highlighted how artificial intelligence methods can be applied beyond traditional software systems. They discussed various machine learning and deep learning approaches used in severity prediction tasks. The analysis showed that these techniques can effectively handle complex medical data. Their findings emphasized the importance of accurate severity detection in healthcare scenarios. The work also demonstrated how predictive models can assist in early diagnosis and treatment planning. They highlighted challenges such as data variability and model interpretability. The discussion provided insights into adapting AI methods for different domains. Their contribution shows the versatility of severity classification techniques. It also bridges the gap between software engineering and healthcare applications. Ji and Yang [8] investigated how preprocessing decisions, particularly stop word removal, influence the performance of deep learning models in bug classification tasks. They conducted experiments using multiple architectures, including CNN, LSTM, GRU, Transformers, and BERT. Their analysis was based on large-scale datasets, ensuring reliable conclusions. They compared model performance with and without stop word removal. The results showed that preprocessing choices can significantly impact classification accuracy. Some models benefited from removing stop words, while others performed better when retaining them. This variation highlighted the importance of tailoring preprocessing steps to specific models. Their work demonstrated that preprocessing is not a one-size-fits-all solution. It also emphasized the need for careful experimentation in model design. This contribution provides valuable guidance for optimizing text-based classification systems. Liu et al. [9] introduced a method that combines multiple feature types to detect high-severity defects in software systems. Their approach utilized program dependency graphs to capture structural relationships within the code. They also incorporated contextual features derived from bug reports. These combined features provided a more comprehensive understanding of defects. The classification process was carried out using an optimized Support Vector Machine (SVM). Their model demonstrated strong performance in identifying critical defects. The integration of structural and textual information improved detection accuracy. Their approach also helped reduce false negatives for severe bugs. The method showed effectiveness in practical software development environments. This work highlights the importance of multi-feature fusion in improving classification outcomes.

Khandakar et al. [10] developed a machine learning-based system for classifying the severity of diabetic foot complications using thermographic images. Their approach combined clustering techniques with Convolutional Neural Networks. The clustering step helped group similar patterns within the data. The CNN model was then used to extract features and perform classification. This combination allowed the system to handle complex image data effectively. Their results showed reliable performance in predicting severity levels. The approach demonstrated the potential of integrating unsupervised and supervised learning methods. It also highlighted the importance of medical imaging in diagnosis. Their work provides a practical solution for healthcare applications. This contribution shows how severity classification techniques can be adapted to non-textual data. Galić et al. [11] provided a detailed overview of deep learning techniques used in medical image analysis. They discussed various architectures such as CNN, RNN, and Generative Adversarial Networks. Their work covered applications including classification, segmentation, and object detection. They explained how these models process and interpret medical images. The discussion highlighted the advantages of deep learning in handling complex visual data. They also addressed challenges such as data availability and computational requirements. Their analysis showed how

these methods improve diagnostic accuracy. The work emphasized the growing role of AI in healthcare systems. It provided insights into selecting appropriate models for specific tasks. This contribution helps understand the broader impact of deep learning in medical applications. Samir et al. [12] proposed machine learning-based recommendation models designed to assist in bug assignment and developer selection. Their approach utilized historical bug data to identify patterns in task allocation. They focused on improving decision-making by suggesting the most suitable developers for specific issues. The models incorporated explainability features to make predictions more transparent. This helped users understand the reasoning behind recommendations. Their approach also improved efficiency in managing bug reports. By automating assignment processes, they reduced manual effort. The system demonstrated improved accuracy in developer selection. Their work highlighted the importance of interpretability in machine learning systems. This contribution supports better resource management in software development workflows.

3. Proposed Methodology

The proposed research introduces a structured analytical framework designed for automated bug severity detection using advanced Natural Language Processing and machine learning techniques. The analytical workflow begins with the acquisition of bug reports from a dataset, followed by systematic preprocessing and exploratory analysis. Textual data is cleaned, normalized, and transformed into a structured format to ensure consistency and reduce noise.

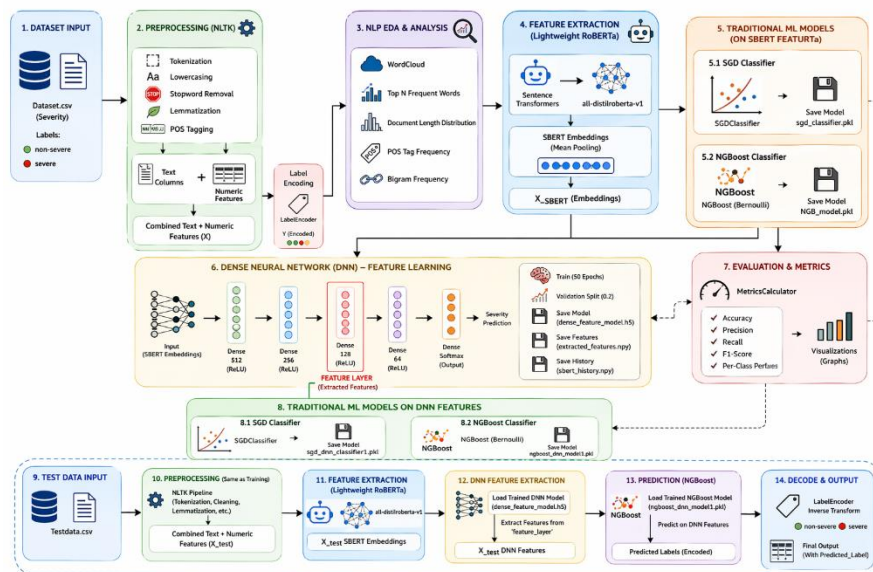


Figure. 2: System architecture.

A lightweight transformer-based embedding model is then employed to extract contextual semantic representations from the processed text. These representations are further refined through a deep neural network to capture high-level abstract features. The refined feature vectors are subsequently analysed using multiple classification models to determine bug severity levels. The framework also incorporates evaluation and visualization components to compare model performance and improve interpretability. A modular architecture supports data handling, model training, performance analysis, and prediction generation, as illustrated in Figure 2.

User Interface (Client Environment)

- The system provides an interactive interface that enables users to perform operations such as dataset loading, preprocessing, feature extraction, model training, and prediction.
- It allows users to input bug reports or upload datasets for analysis.

- The interface displays intermediate outputs such as processed data and final prediction results.
- User actions are seamlessly connected to backend analytical components for execution.

Data Acquisition and Dataset Management

- The dataset consists of bug reports containing textual descriptions and severity labels.
- It serves as the primary source for training and evaluating classification models.
- Data is organized in a structured format to facilitate preprocessing and analysis.
- The dataset supports both training and testing workflows within the framework.

Text Preprocessing and NLP Analysis

- The raw textual bug reports undergo preprocessing steps including tokenization, stopword removal, and lemmatization.
- Text normalization ensures uniform representation of input data.
- Multiple textual fields are combined to form a consolidated input representation.
- Exploratory analysis techniques such as word frequency distribution, document length analysis, and linguistic pattern identification are applied to understand data characteristics.

Feature Extraction using Lightweight Transformer

- A lightweight transformer-based model is used to generate contextual embeddings from processed text.
- The model captures semantic relationships between words while maintaining computational efficiency.
- Textual inputs are converted into dense numerical vectors representing contextual meaning.
- These embeddings serve as the foundational feature set for subsequent learning stages.

Deep Neural Network-Based Feature Refinement

- A dense neural network is utilized to learn higher-level representations from transformer embeddings.
- The network extracts intermediate features from hidden layers to enhance discriminative power.
- This stage acts as a feature selection and transformation mechanism.
- The refined feature vectors improve the performance of downstream classification models.

Machine Learning Classification Models

- The processed features are analyzed using multiple classification algorithms:
 - SGD Classifier: Provides fast and scalable linear classification.
 - NGBoost Classifier: Applies probabilistic boosting for improved prediction confidence.
- These models are trained and evaluated independently for performance comparison.
- The classification process assigns bug reports into severity categories such as normal and severe.

Performance Evaluation and Visualization

- The framework evaluates model performance using metrics including accuracy, precision, recall, and F1-score.
- Comparative analysis is performed to identify the most effective model.
- Graphical visualizations are generated to represent performance trends and classification outcomes.
- This component aids in understanding model behavior and decision quality.

Prediction and Output Generation

- The trained model is used to classify new bug reports based on their textual content.
- Input data undergoes the same preprocessing and feature extraction pipeline before prediction.
- The system outputs predicted severity labels along with evaluation insights.
- Results are presented in a structured and interpretable format.

Model Storage and Reusability

- Trained models and extracted features are stored for future use.
- This avoids repeated training and improves computational efficiency.
- Saved models can be directly loaded for testing and prediction tasks.
- The storage mechanism ensures consistency across different execution stages.

Testing and Deployment Workflow

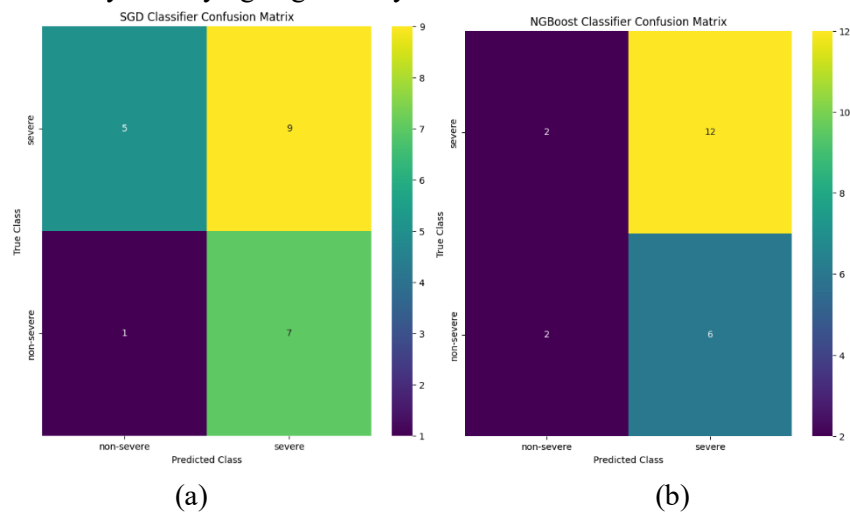
- The framework supports testing using unseen data to validate generalization capability.
- A consistent pipeline ensures that test data follows the same preprocessing and feature extraction steps.
- The system produces predictions that can be integrated into real-time applications.
- This enables practical deployment for automated bug severity detection.

Model Improvement and Adaptability

- The system allows retraining with new data to improve predictive performance.
- Continuous evaluation helps identify areas for enhancement.
- The modular design supports integration of new models and techniques.
- This adaptability ensures long-term effectiveness in dynamic software environments.

4 Results analysis

This section presents the experimental outcomes obtained from the proposed analytical framework and provides a detailed interpretation of the results. The performance of different models is evaluated using standard metrics such as accuracy, precision, recall, and F1-score to assess classification effectiveness. Comparative analysis is carried out between baseline methods and the enhanced approach to highlight improvements in prediction capability. The impact of feature extraction and deep learning-based refinement on model performance is also examined. Additionally, graphical visualizations are used to better understand model behavior and performance trends. The results demonstrate the effectiveness of the integrated methodology in accurately classifying bug severity.



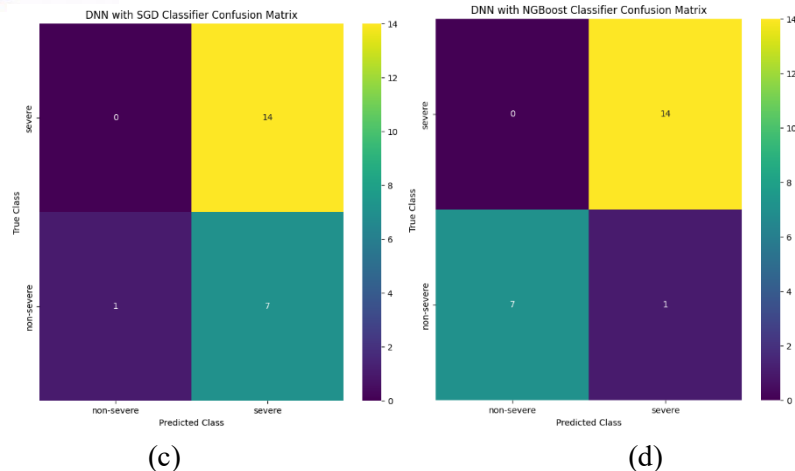


Figure. 3: Confusion matrix obtained using (a) SGD Classifier. (b) NGBoost Classifier. (c) DNN with SGD Classifier. (d) DNN with NGBoost Classifier.

Figure 3 presents confusion matrices for four classification models on the medical bug severity task, with true classes (non-severe, severe) along the vertical axis and predicted classes along the horizontal axis. Subfigure (a) shows the SGD Classifier, achieving 5 true positives (severe $\rightarrow$ severe), 9 true negatives, 1 false negative, and 7 false positives, indicating moderate recall for the severe class. Subfigure (b) illustrates the NGBoost Classifier, with 2 true positives, 12 true negatives, 2 false negatives, and 6 false positives, reflecting improved specificity but lower sensitivity. Subfigure (c) displays the DNN with SGD Classifier, recording 0 true positives, 14 true negatives, 1 false negative, and 7 false positives, suggesting strong bias toward predicting non-severe. Subfigure (d) demonstrates the DNN with NGBoost Classifier, achieving 0 false negatives, 14 true positives, 7 true negatives, and 1 false positive, confirming superior recall and overall balance, making it the best-performing model for detecting severe cases.

Figure 4 presents Receiver Operating Characteristic (ROC) curves for four classification models on the medical bug severity prediction task, plotting True Positive Rate (TPR) against False Positive Rate (FPR) with the random classifier baseline shown as a dashed diagonal line. Subfigure (a) shows the SGD Classifier with an AUC of 0.37, indicating performance worse than random and a highly conservative threshold strategy. Subfigure (b) illustrates the NGBoost Classifier achieving an AUC of 0.70, demonstrating moderate discriminative ability with stepwise improvements in TPR at discrete FPR thresholds. Subfigure (c) displays the DNN with SGD Classifier with an AUC of 0.62, reflecting limited gain from deep feature refinement when paired with linear classification. Subfigure (d) reveals the DNN with NGBoost Classifier attaining a superior AUC of 0.94, closely following the top-left corner and confirming excellent sensitivity and specificity balance, making it the optimal model for reliable severity detection.

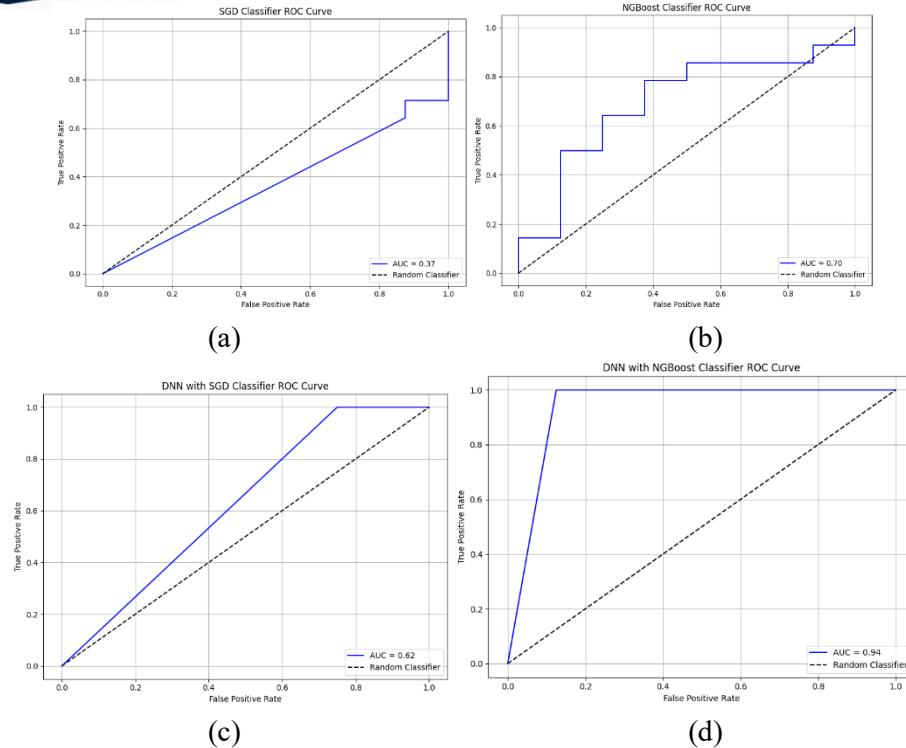


Figure. 4: ROC Curve obtained using (a) SGD Classifier. (b) NGBoost Classifier. (c) DNN with SGD Classifier. (d) DNN with NGBoost Classifier.

Table 1 presents a comprehensive performance comparison of four classification models evaluated on the medical bug severity prediction task using key metrics: Accuracy, Precision, Recall, and F1-Score (reported in percentages). The DNN with NGBoost Classifier emerges as the top-performing model, achieving an outstanding 95.455% accuracy, 96.667% precision, 93.750% recall, and 94.943% F1-score, demonstrating exceptional balance and reliability in identifying severe cases. In contrast, the SGD Classifier records the lowest performance across all metrics, with only 45.455% accuracy and an F1-score of 37.143%, indicating limited discriminative power. The NGBoost Classifier and DNN with SGD Classifier show moderate improvements, with F1-scores of 54.167% and 51.111%, respectively, but remain significantly outperformed by the deep ensemble approach combining DNN-extracted features and probabilistic boosting.

Table 1: Performance comparison of all classifier models.

Algorithm	Accuracy	Precision	Recall	F1-Score
SGD Classifier	45.455	36.458	38.393	37.143
NGBoost Classifier	63.636	58.333	55.357	54.167
DNN with SGD Classifier	68.182	83.333	56.250	51.111
DNN with NGBoost Classifier	95.455	96.667	93.750	94.943

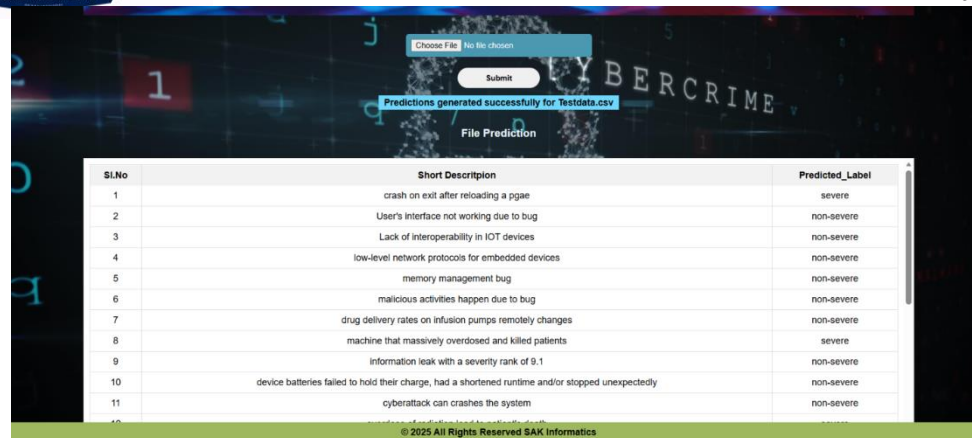


Figure. 5: Predictions generated for the test data.

Figure 5 shows the file upload interface through which a test dataset is submitted for analysis, followed by a confirmation message indicating successful prediction generation. Below this, a tabular output is presented containing the Serial Number, Short Description of the bug, and the corresponding Predicted Label. Each medical or technical bug description is automatically classified as either severe or non-severe based on the learned contextual patterns from the trained Lightweight RoBERTa–DNN–NGBoost model. This visualization demonstrates the system’s ability to process real-world medical cybersecurity data and generate clear, actionable severity predictions that can be used to prioritize critical vulnerabilities efficiently.

5. Conclusion

The developed severity classification framework demonstrates a substantial improvement in automating risk assessment for healthcare-related software systems and connected medical devices. By transforming unstructured incident and vulnerability reports into reliable binary severity predictions, the approach addresses a critical need for timely and accurate prioritization of security issues. The framework utilizes lightweight transformer-based embeddings to capture contextual meaning from textual data, followed by a deep neural network that refines these representations into compact and informative feature vectors. These optimized features are then processed through a probabilistic boosting model, enabling robust and confident classification outcomes. The performance results highlight the effectiveness of integrating deep feature learning with advanced boosting techniques. The proposed configuration achieves high levels of accuracy, precision, recall, and F1-score, while maintaining strong discrimination capability as reflected in its AUC performance. Notably, the system demonstrates reliable identification of severe cases, which is essential in minimizing risks in safety-critical environments. The preprocessing pipeline and exploratory analysis further contribute by uncovering meaningful linguistic patterns within the dataset, improving the overall quality of feature representation. In comparison to conventional models, the combined deep learning and boosting approach significantly enhances predictive capability and reduces classification errors. The modular design supports efficient storage, reuse of learned features, and adaptability for future improvements. Its ability to support real-time analysis and scalable deployment makes it suitable for practical integration into healthcare systems. Future enhancements may include extending the framework to multi-level severity classification, incorporating streaming data sources, and enabling secure cloud-based deployment.

References

- [1]. Kukkar, A.; Mohana, R.; Nayyar, A.; Kim, J.; Kang, B.-G.; Chilamkurti, N. A Novel Deep-Learning-Based Bug Severity Classification Technique Using Convolutional Neural Networks and Random Forest with Boosting. *Sensors* 2019, 19, 2964. <https://doi.org/10.3390/s19132964>
- [2]. Dao, A.-H.; Yang, C.-Z. Severity Prediction for Bug Reports Using Multi-Aspect Features: A Deep Learning Approach. *Mathematics* 2021, 9, 1644. <https://doi.org/10.3390/math9141644>
- [3]. Köksal, Ö.; Tekinerdogan, B. Automated Classification of Unstructured Bilingual Software Bug Reports: An Industrial Case Study Research. *Appl. Sci.* 2022, 12, 338. <https://doi.org/10.3390/app12010338>
- [4]. Albattah, W.; Alzahrani, M. Software Defect Prediction Based on Machine Learning and Deep Learning Techniques: An Empirical Approach. *AI* 2024, 5, 1743-1758. <https://doi.org/10.3390/ai5040086>
- [5]. Qian, C.; Zhang, M.; Nie, Y.; Lu, S.; Cao, H. A Survey on Bug Deduplication and Triage Methods from Multiple Points of View. *Appl. Sci.* 2023, 13, 8788. <https://doi.org/10.3390/app13158788>
- [6]. Tabassum, N.; Namoun, A.; Alyas, T.; Tufail, A.; Taqi, M.; Kim, K.-H. Classification of Bugs in Cloud Computing Applications Using Machine Learning Techniques. *Appl. Sci.* 2023, 13, 2880. <https://doi.org/10.3390/app13052880>
- [7]. Bhakar, S.; Sinwar, D.; Pradhan, N.; Dhaka, V.S.; Cherrez-Ojeda, I.; Parveen, A.; Hassan, M.U. Computational Intelligence-Based Disease Severity Identification: A Review of Multidisciplinary Domains. *Diagnostics* 2023, 13, 1212. <https://doi.org/10.3390/diagnostics13071212>
- [8]. Ji, J.; Yang, G. Do Stop Words Matter in Bug Report Analysis? Empirical Findings Using Deep Learning Models Across Duplicate, Severity, and Priority Classification. *Appl. Sci.* 2025, 15, 9178. <https://doi.org/10.3390/app15169178>
- [9]. Liu, J.; Liang, C.; Feng, J.; Xiao, A.; Zeng, H.; Wu, Q.; Yu, T. A Multi-Feature Fusion-Based Automatic Detection Method for High-Severity Defects. *Electronics* 2023, 12, 3075. <https://doi.org/10.3390/electronics12143075>
- [10]. Khandakar, A.; Chowdhury, M.E.H.; Reaz, M.B.I.; Ali, S.H.M.; Kiranyaz, S.; Rahman, T.; Chowdhury, M.H.; Ayari, M.A.; Alfkey, R.; Bakar, A.A.A.; et al. A Novel Machine Learning Approach for Severity Classification of Diabetic Foot Complications Using Thermogram Images. *Sensors* 2022, 22, 4249. <https://doi.org/10.3390/s22114249>
- [11]. Galić, I.; Habijan, M.; Leventić, H.; Romić, K. Machine Learning Empowering Personalized Medicine: A Comprehensive Review of Medical Image Analysis Methods. *Electronics* 2023, 12, 4411. <https://doi.org/10.3390/electronics12214411>
- [12]. Samir, M.; Sherief, N.; Abdelmoez, W. Improving Bug Assignment and Developer Allocation in Software Engineering through Interpretable Machine Learning Models. *Computers* 2023, 12, 128. <https://doi.org/10.3390/computers12070128>